A Close Look at the Compact XML Schema-Aware XML Processing Framework

Jinyu **Wang**

Abstract

Wide deployment of XML technology in enterprise applications demands high performance XML processing framework. This results in extensive investigation on building an XML processing infrastructure leveraging a compact, pre-parsed XML format, which could save in the memory and CPU consumption as well as the network bandwidth.

In this paper, we will discuss the project building a compact schema-aware binary XML processing framework and compare it with the existing binary XML technologies. The discussion will cover the design of the compact binary XML format, the implementation for the compact binary XML processors, which encode and decode the XML documents, and how the compact binary XML support is integrated with the existing XML processing stack.

At the end, we will provide the result testing applications leveraging the compact binary XML processing framework.

Table of Contents

1. Introduction

As XML becomes the basis for the e-commerce, Web services and integration applications, the challenge is on providing an XML processing infrastructure handling large volume of XML data and delivering the enterprise required performance. This results in extensive investigation on building an XML processing infrastructure leveraging a compact, pre-parsed binary XML format, which could save in the memory and CPU consumption as well as the network bandwidth. Several motivations drive this effort.

First of all, text-based XML data is verbose. Every XML element requires both a start tag and an end tag. In certain cases, XML tags could take more space than the amount of space used by the actual data. With a binary XML format replacing XML tags with the token ids, XML documents will have a much more compact size.

Second, text-based XML data doesn't aware of data types. Keeping data types such as float numbers or dates in text requires more storage space. Additionally, data type conversion has to be done during the XML parsing. With binary XML, data can be stored in its native format to reduce storage space as well as the time and effort necessary to conduct the data type conversions.

Third, enterprise applications generally require a multi-tier collaboration. Sharing XML data in text results in multiple XML generation and parsing steps. The XML processing result in one step can't be used by the next step. With binary XML, the processing results can be included along with the data. Therefore, the overall application processing will be more efficient.

In this paper, we will give details on a project building a compact schema-aware binary XML processing framework. The project is based on several years investigation and practices on binary XML support.

In Oracle9i, Oracle XDK introduced a binary XML format (CXML), which tokenized the XML metatags and providing them as inline tokens with the binary XML streams. A similar approach is used by ASN.1 (Fast Infoset). This approach is successful because it reduces the XML file size and provides quick binary data serilization/deserlization. However, without leveraging XML schemas, the format lacks of data type awareness thus limits its capability to optimize the compression. The format is also not extensible to carry the application context for multi-tier applications.

Comparing with another popular binary XML initiative--XOP (XML Binary Optimization Packaging)-- the project doesn't limit to dealing with binary XML data in XML documents. Instead, the proposed binary XML solution provides a format that compresses the entire XML document while preserving the document structure information.

In the following sections, we will extent the discussion how the binary XML processing framework is built:

Section 2 looks at the business use cases for compact binary XML including XML database management, Web services, and BEPL integration applications.

Section 3 discusses the design tradeoffs for the compact binary XML format.

Section 4 covers the design of the compact binary XML processing infrastructure

Section 5 shows how to integrate the compact binary XML support with the existing XML processing stack.

Finally, we conclude by providing the result testing applications leveraging the compact binary XML processing framework.

2. Business Use Cases

In order to properly address the requirements on the compact bianary XML support, the appropriate use cases are collected. In the project the following use cases are taken into consideration.

Use Case 1: XML DB Storage Optimization

XML DB is a repository where XML documents are stored, searched and queried. The existing Oracle XML DB storage is either on LOBs or object-relational tables. LOB storage is slow to process because it requires XML parsing before any operations. LOB storage is also not data type awared and provides limited indexability. To efficiently process XML data, Oracle XML DB uses object-relational (O-R) storage. O-R storage preparses the XML document and stores the data in boject-relational tables. The storage speeds up the data access. However, it is not flexibility when the XML schema used to define the object-relational tables evolves. Therefore, XML DB needs a new binary XML format to store XML Data for quick data access and simplify the data management when XML schemas evolve. Additionally, by supporting the same binary XML format in mid-tier, mid-tier applications can directly use the XML data that is parsed, indexed and validated by XML DB.

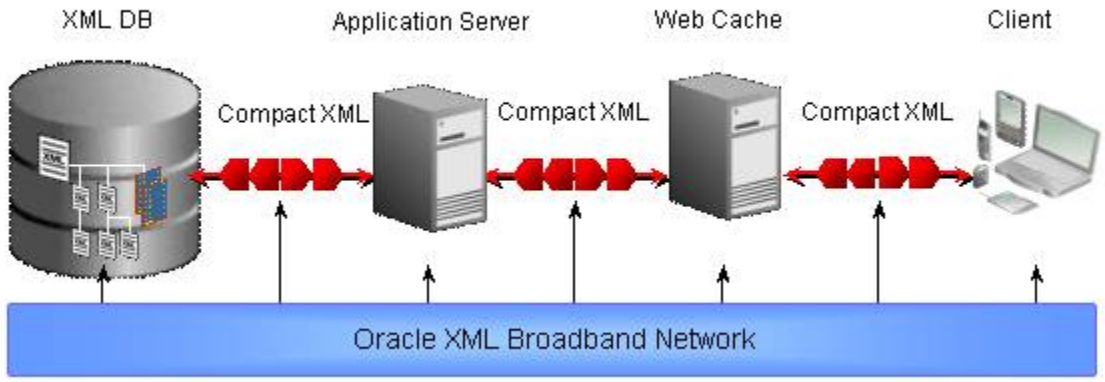

Figure 1. Multi-tier Compact Binary XML Support

Figure-1 shows an example multi-tier XML application delivering the content to the Web clients. The application retrieves XML data from the database, applies the business logic in the application server and caches the data within the Web cache. Supporting thousands of concurrent users, the application has to process large volume of XML data every second. If XML is shared across tiers in text, not only the size of the data test the system to its limit, XML parsing in every tier also slows down the process. There is almost no cooperation between the appliations. With binary XML data, XML can be parsed once and used multiple times. The cross-tier data processing becomes possible.

Use Case 2: BPEL Dehydration

In BPEL applications, BPEL processes can be asynchronous and certain process may take long-time waiting for the partner responses. Keeping the process data in memory is not efficient. Therefore, a dehydration process is introduced to write the cached process data in XML into a persistent storage, thus freeing up server resources, as shown in Figure -2.

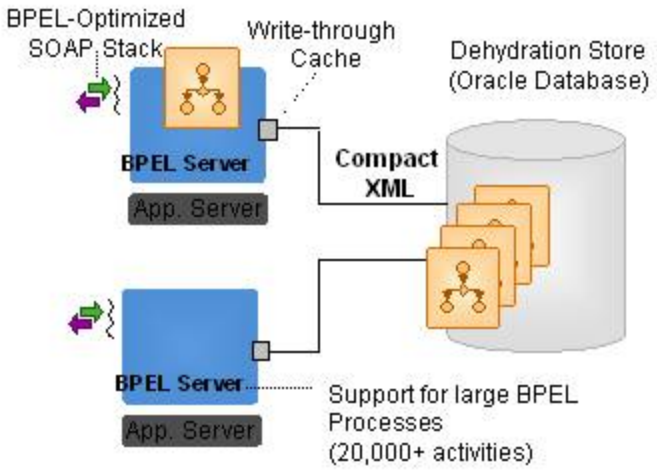

Figure 2. Oracle BPEL Dehydration using the Compact Binary XML

By storing the data in binary XML, BPEL servers can quickly serialize/deserialize binary XML data and support large XML documents. By supporting smart partitioning of the binary XML data during serialization, the BPEL servers can also lazily load and save data.

Use Case 3: Web Services

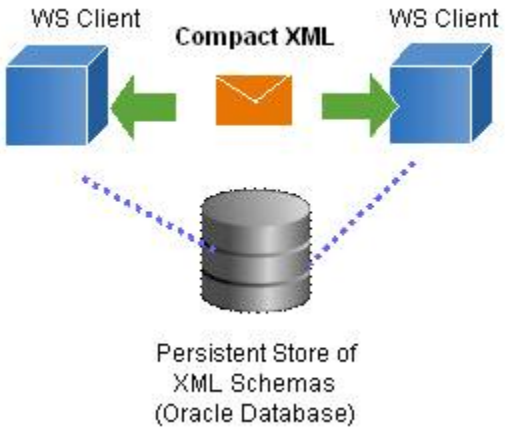

Figure 3. Web Service Using Binary XML as the Message Payload

Figure-3 shows the use case where Web services using binary XML as the message payload. Using binary XML, Web service applications can reduce the network payload and memory footprint. In this use case, non-schema based bianry XML format is suggested because all the bianry tokens are kept inline with the data. When decoding a non-schema based bianry XML document, the decoding client doesn't have to know any server information. While sharing a schema-based binary XML data, there is a requirement on decoding clients to access a persistent storage which stores the XML schemas used for the binary data serialization.

3. Designing the Compact Binary XML Format

Based on the describe use case, a compact binary XML format is designed to preserve the infoset fidelity and allow efficiently storing and manipulating XML data within databases as well as transferring between applications. However, there is no one-size-fits-all solution. Several design decisions have to be made.

First of all, rather than optimized for building content indexes, the format is designed for efficiently serializing and deserializing XML data. The format preserves the XML document order, which then allows applications to serialize and deserialize XML data in stream-based processes. This is because content indexes are tied to application use cases while any particular application might want to build its own indexes. For this reason, such index data should typically kept outside of the data itself, such as in a database.

Second, in order to facilitate real-time content delivery across tires, the format is designed for delivering a high processing speed rather than a high compression ratio. The emphasis is on producing SAX/StaX events from the binary XML data. Some compression is made such as the XML tag compression and the XML schema-based text compression. No further compression is allowed because in order to have high compress ratio, the process has to optimize the tree evaluation, which decreases the processing speed. In other words, the compression is made only if it improves the processing speed. This means the compression is not optimized.

Third, the compact binary XML format is designed to be able to leverage the metadata in the XML schemas or the external metadata information such as the database metadata information. The use of XML schemas will reduce the tag overhead and make use of the structure information for compression optimization. The advantage of this will be the compression optimization. However, it depends on the availability of the XML schemas.

Fourth, the compact binary XML format is designed to support random data access, which means queries such as the XPath queries may point into the middle of the document. In order to retrieve the data efficiently, the binary XML format must keep the state of process small and push down the namespace and declarations information to the XML element level.

Finally, though the compact binary XML data is compressed, since the design is focus on the processing speed, the size of document can still be large. To support large XML documents, the compact binary format is designed to support lazy loading, which means that the data can be broken into smaller chucks, and loaded to memory only if the processing needs it.

With all these design considerations, the compact binary XML is a format which enables streamable XML processing and optimized for random access.

4. Building a Compact Binary XML Processing Infrastructure

Figure -4 shows a compact binary XML processing infrastructure which provides both database and mid-tier support. In both tiers, compact binary XML processors are provided to encode and decode binary XML data. For schema-awared binary XML, Oracle databases uses XML DB repository to manage the XML schemas, while mid-tier compact binary XML processors allows applications to plug-in a persistent storage.

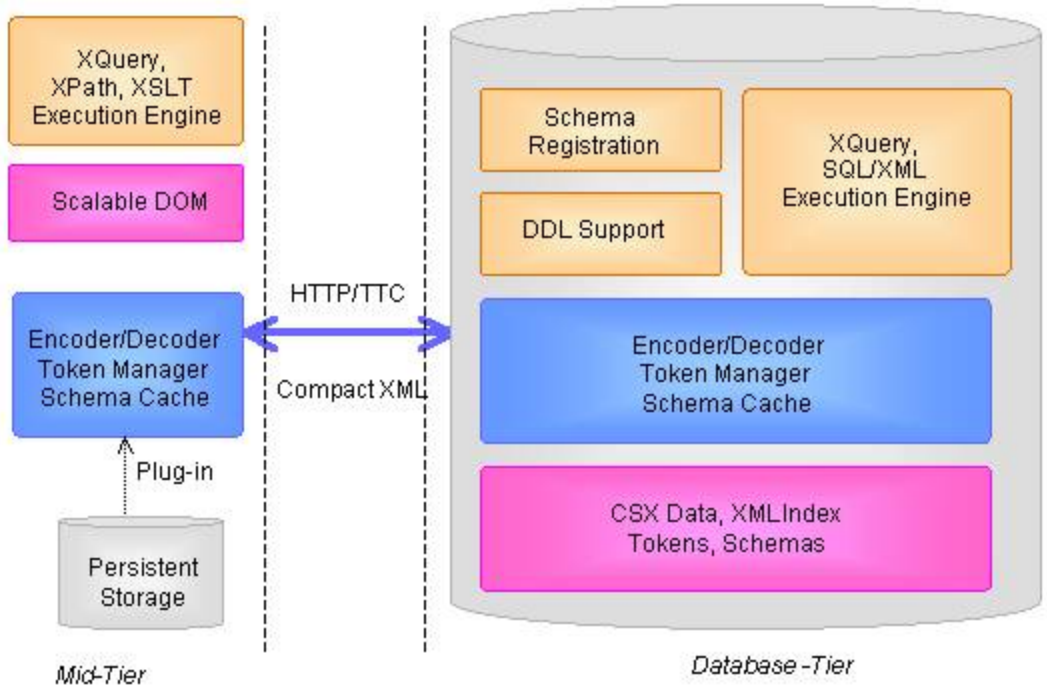

Figure 4. XML Processing Infrastructure

5. Integrating the Compact Binary XML Data

In order to allow existing XML components such as XPath, XSLT and XQuery processors to support the binary XML data with minimum implementation changes, a standard-based utility called scalable DOM is created. This utility extends the existing DOM implementation and provides APIs for plugging in the new XML data format.

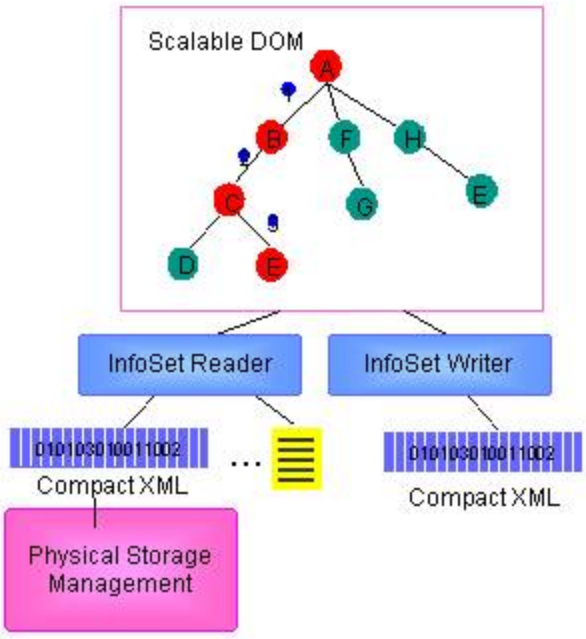

Figure 5. Scalable DOM Architecture

Figure-5 shows the architecture of the scalable DOM implementation, which splits the DOM implementation into two separate layers: the DOM API layer and the data layer. The DOM API layer consists of a lightweight and configurable DOM-based API supporting node access, navigation, update and search. The data layer provides standard InfoSetReader and InfosetWriter APIs for plugging in various data sources. Because the DOM layer allows applications access and update XML data regardless of the actual data format, XML components built on scalable DOM can support any plug-in data including the compact binary XML data without any changes.

6. Conclusion

Compact binary XML data gives the XML processing infrastructure more flexibility to improve the scalability and performance. With binary XML data stored, Oracle XML DB is tested to be able to reduce the storage of a 11MB XML document down to 8 MB. A mid-tier application directly leveraging the binary XML data from XML DB without reparsing the data can reduce the processing time by 50%. The future plan is to test the compact binary XML support on end-to-end applications.

Acknowledgements

Thanks to Eric Sedlar, Kongyi Zhou, Meghna Mehta, Nipun Agarwal and Sivasankaran Chandrasekar, Bhushan Khaladkar and all the other people in the Oracle XML Development team for the help on this paper.

Bibliography

[Dmitry03] *Binary XML, W3C workshop on Binary Interchange of XML Information Item Sets, 2003*, Demitry Lenkov et. al.

Biography

Jinyu **Wang**
Senior Product Manager
Oracle Corporation [http://www.oracle.com]
Redwood Shores
California
United States of America

Jinyu Wang is a senior product manager for Oracle XML Product management in charge of building the XML infrastructure for all Oracle product stacks and providing the enterprise XML content management support. Jinyu also works on providing XML solutions for Oracle customers and evangelizing Oracle XML solutions. She is a co-author of Oracle Database 10g: XML and SQL. She got her master's degree in computer science at University of Southern California.